Google Summer of Code 2025 Proposal

Project: Developing 8 Math games for Sugar

Basic Info

Name	Ali Hassan
Github	AliHassan245
Email	alihsn.24.5.3@gmail.com
College	National University of Science & Technology (NUST) Pakistan
Major/Minor	BS Electrical Engineering / Artificial Intelligence
Languages	English, Urdu (Proficient in Writing and Speaking)
Location / Time zone:	Karachi, Pakistan Standard Time (GMT+5)

About Me

Introduction

I am a second-year BS Electrical Engineering student specializing in Artificial Intelligence (AI). I have a keen passion for web development, machine learning, mathematics, and game development. Additionally, I have a strong interest in contributing to open-source projects to drive innovative solutions and make a meaningful impact in the tech community.

Previous Open Source Experience

In addition to that, I have been working as an AI engineer in the Robotics and AI Society at my college, where we built an AI-powered humanoid robot (MARC-I). In this project, my contributions focused on implementing computer vision for real-time object detection and recognition, along with reinforcement learning to enable the robot to adapt its actions based on environmental feedback, allowing it to perceive its surroundings and make intelligent, data-driven decisions.

Society Work

In addition to that, I have been working as an AI engineer in the Robotics and AI Society at my college, where we built an AI-powered humanoid robot (MARC-I). In this project, my contributions focused on implementing computer vision for real-time object detection and recognition, along with reinforcement learning to enable the robot to adapt its actions based on environmental feedback, allowing it to perceive its surroundings and make intelligent, data-driven decisions.

Past Contributions

S.No.	Activity Name	Issues
1.	**3D Volume**	**https://github.com/llaske/sugarizer/issues/1774**
2.	**Color My World**	**https://github.com/llaske/sugarizer/issues/1771**
3.	**Chart**	**https://github.com/llaske/sugarizer/issues/1727**
4.	**Moon**	**https://github.com/llaske/sugarizer/issues/1730**
5.	**Block Rain**	**https://github.com/llaske/sugarizer/issues/1739**
6.	**Abecedarium**	**https://github.com/llaske/sugarizer/issues/1734**
7.	**Food Chain**	**https://github.com/llaske/sugarizer/issues/1756**

S.No.	Activity Name	Pull Requests
1.	**3D Volume**	**https://github.com/llaske/sugarizer/pull/1780**
2.	**E-Book**	**https://github.com/llaske/sugarizer/pull/1769**
3.	**Color My World**	**https://github.com/llaske/sugarizer/pull/1773**
4.	**Moon**	**https://github.com/llaske/sugarizer/pull/1732**
5.	**Chart**	**https://github.com/llaske/sugarizer/pull/1729**

Personal Technology Stack

- **Programming Languages**: Python, C/C++, Java, JavaScript
- **Game Development**: PhaserJS, ThreeJS
- **Web Development**: HTML, CSS, Bootstrap, Express.js, EJS, JQuery, React & Vue.js, Node.js, Fast API, Rest API, Postman, MongoDB & PostgreSQL.
- **Machine Learning(AI)**: Scikit-Learn, NumPy, Pandas
- **Computer Vision**: OpenCV
- **Embedded & Robotics**: Arduino, ROS2
- **Design & Prototyping**: Figma, Adobe Photoshop
- **Version Control**: Git and GitHub
- **Mathematics**: Linear Algebra, (ODEs), Multivariable Calculus, Probability & Statistics
- **Mathematical & Circuit Simulation**: MATLAB, Multisim
- **Operating Systems & Development**: Linux, Windows

Why Sugar Labs?

I was looking for GSoC organizations when I discovered Sugar Labs and was inspired by how they are revolutionizing education by integrating technology into learning. As I explored their repositories and projects, I was particularly impressed by their use of **Artificial Intelligence and Game Development** as educational tools. As a developer, I became even more interested in contributing to Sugar Labs because it perfectly aligns with my love for **Mathematics, AI, and Game Development**. This motivated me to start contributing to various Sugar activities and games.

Another thing I liked about Sugar Labs is their mission to provide **open-source educational tools and fun activities for kids**. I'm looking forward to **continuing** my contributions to Sugar Labs and making a positive impact on education through technology.

Project Goal

I plan to develop **at least eight math games** for Sugar Labs during the **GSoC 2025** timeline, making learning **engaging and interactive** for children. These activities will include **classic puzzles** like **Four Color Map Game, Fifteen Puzzle, and Euclid's Game**, alongside **AI-powered games** such as **Number Detective** and **Sorting Hat AI**, which introduce kids to **machine learning concepts**. The games will feature **self-designed visual assets** created using **Adobe Photoshop**, ensuring an intuitive and visually appealing experience. Development will focus on **core gameplay functionalities**, adhering to **Sugar Labs' open-source principles**, and ensuring seamless integration with the **Sugar desktop environment**.

How will it Impact SugarLabs

Upon completing this project, **Sugar Labs will gain eight new, fully developed math activities**. These games, including **Four Color Map Game, Fifteen Puzzle, Euclid’s Game, Number Detective, Sorting Hat AI, and others**, will introduce children to **core mathematical concepts** such as **logical reasoning, spatial problem-solving, and number patterns** in a **fun and interactive way**. In particular, the **AI-powered games** will provide young learners with an **insight to machine learning concepts** through a **gamified and visually child-friendly manner**.

Project Type

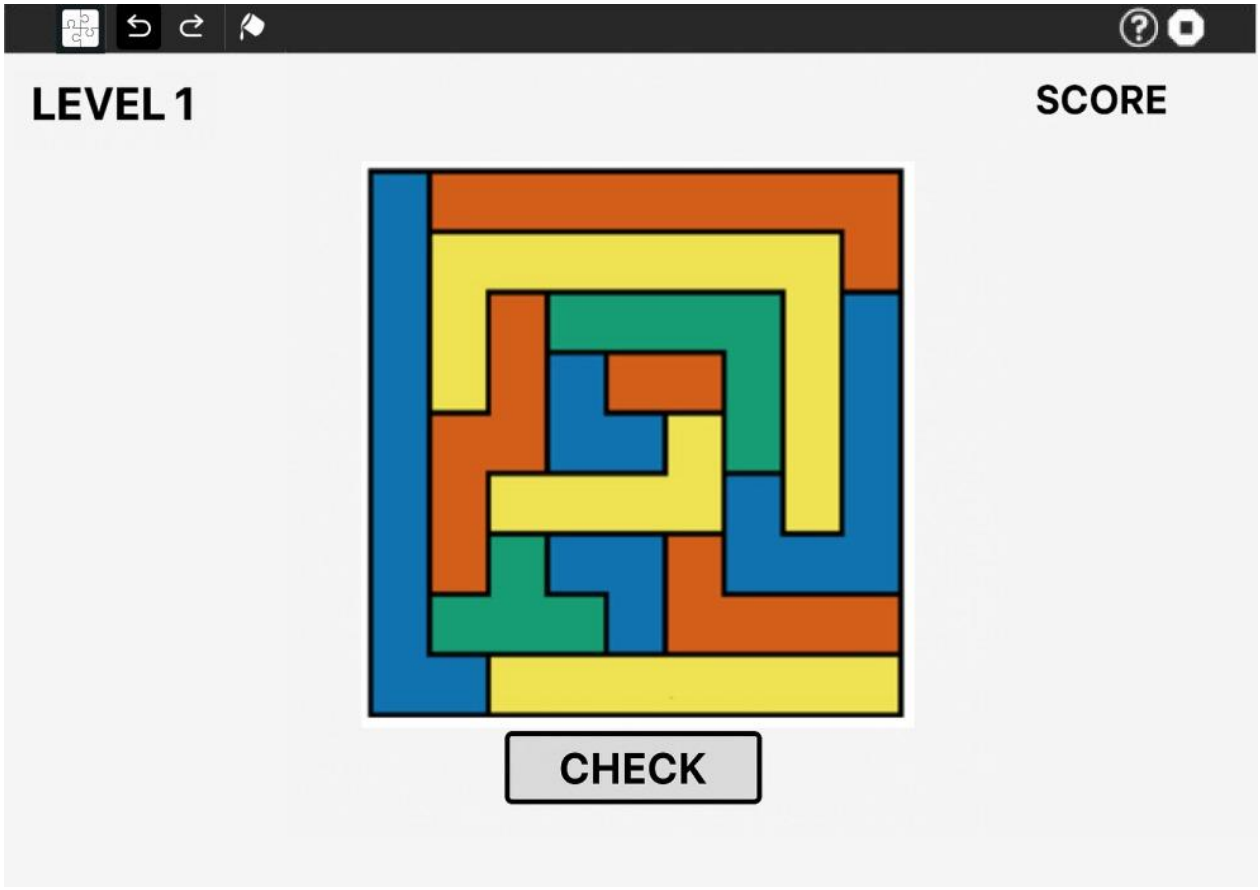
350 hours

Project Details:

Following are the description of the activities I aim to develop:

1. Four Color Theorem

Here is the prototype image of the game, designed by me, showcasing its initial layout and gameplay.



Overview:

This activity is inspired by the classic Four Color Theorem, which states that any arrangement of regions can be colored using only four colors so that no two adjacent regions share the same color. The game employs a dynamic square board filled with shapes, including polygons, irregular forms, and overlapping regions. As levels progress, the board presents different designs that increase the puzzle's complexity.

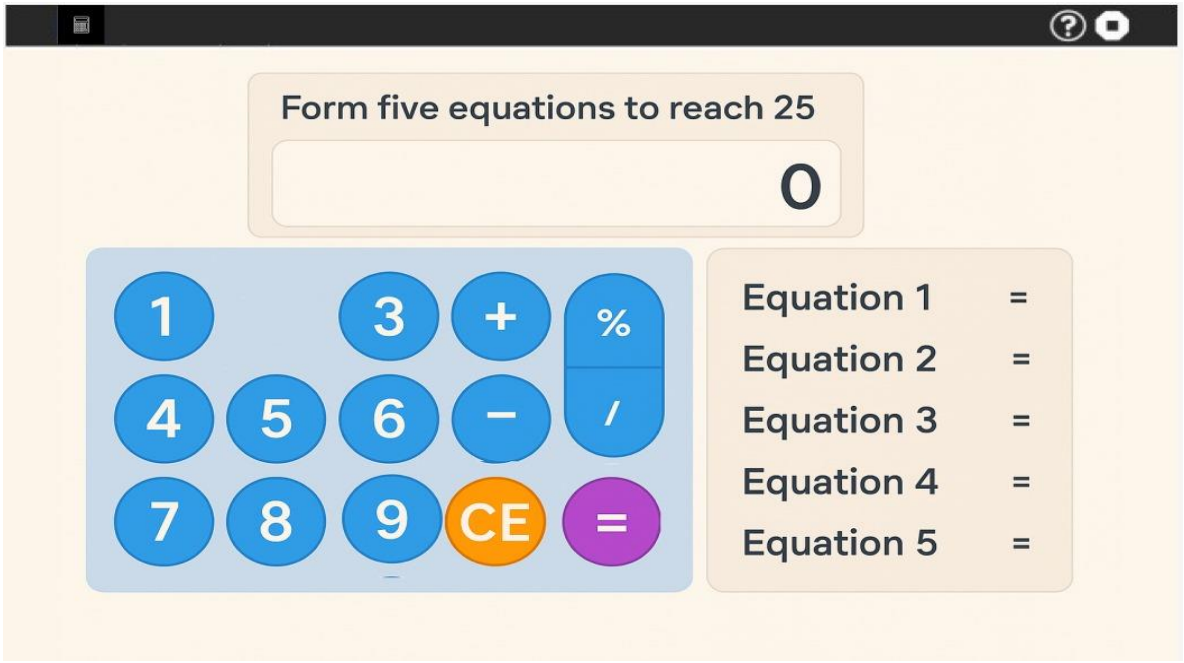
User Interface & Controls:

- **Toolbar:** The interface features a toolbar with essential editing controls including Undo and Redo buttons, alongside a color option offering exactly four distinct colors.
- **Activity Area:** The main area displays the square board filled with shapes. A dedicated Check button is integrated into this area, enabling users to verify their solution by ensuring that no two adjacent shapes in the square board share the same color. Additionally, a level indicator and a score indicator are incorporated to display both the current level and the player's score.

Gameplay Mechanics:

- **Interactive Coloring:** Players select a color from the palette and click on the shapes within the board to apply it.
- **Progressive Difficulty:** The game starts with simpler configurations and gradually introduces more intricate designs as the levels progress. Each new design increases the challenge, requiring more thoughtful color assignments.
- **Solution Verification:** Users can click the Check button in the main activity area to immediately verify their solution.
- **Score Calculation:** The primary basis for calculating the score is move efficiency, players earn higher scores by minimizing the number of moves or color changes needed to complete a level. This system encourages strategic planning and optimal decision-making, rewarding players who complete levels with fewer actions.

2. Broken Calculator



Overview

The Broken Calculator is a number puzzle game where players must form valid mathematical equations to reach a target number, but with a twist that one digit is disabled. This challenge encourages creative problem-solving and arithmetic skills as players navigate around the missing key to construct correct equations.

Game Mechanics

Challenge Setup:

- **Target & Broken Key:** Each round presents a target number. The calculator's key corresponding to the first digit of the target is disabled (e.g., for 25, the "2" key is inactive).
- **Objective:** Players must generate a predefined number of valid equations (for example, five different equations) that evaluate to the target number without using the forbidden digit.

Equation Formation:

- **Allowed Inputs:** Players can use all digits from **0 to 9**, except the **broken key**, along with the four basic arithmetic operators (**+, −, ×, ÷**) to form equations.
- **Equation Submission:** As the broken key is completely removed from the UI, players cannot accidentally use it. They construct each equation via the interactive calculator, and once submitted, it is added to the equation list panel.

Progression and Scoring:

- **Score Indicator:** The UI includes a real-time score indicator that updates after each correctly submitted equation, providing immediate positive reinforcement and a clear sense of progress.
- **Bonus Points:** Optionally bonus points or multipliers can be awarded when a complete set (e.g., all five equations in a round) is correctly formed, encouraging accuracy and efficiency.

User Interface (UI)

- **Toolbar (Top):**
  At the very top of the interface, a toolbar provides quick access to key functions:
  - **Activity Icon:** Navigates to the main game activity.
  - **Stop Option:** Allows users to exit the activity.

- **Main Activity Area (Below the Toolbar):**
  This area is divided into two primary sections and begins with a header:
  - **Header:** Displays the instruction “Form five equations equal to ___” with the target number clearly indicated, ensuring users immediately understand the challenge.
  - **Left Panel – Calculator Interface:**
    • **Digital Display:** The display screen shows the current equation being entered.
    • **Button Grid:** A standard grid layout with buttons for digits and arithmetic operations. The broken key, is completely removed from the interface, so users cannot use it.
    • **Responsive Interactions:** Each button press triggers animations and real-time validations, offering immediate feedback as users build their equation.
  - **Right Panel – Equation List:**
    • **Equation Slots:** This dedicated panel contains five distinct slots labeled “Equation 1” through “Equation 5.”
    • **Dynamic Display:** Each slot initially shows placeholder text and updates to display the submitted equation once entered.
    • **Active Slot Highlight:** Optionally the slot corresponding to the current equation being entered can be highlighted.

3. Soma Cubes

Overview

The Soma Cube game is an interactive 3D puzzle that challenges players to assemble a target shape using the classic seven Soma pieces. Each piece is a unique combination of three or four cubes. The primary challenge is to build the classic 3×3×3 cube in the first level, with subsequent levels featuring more complex shapes (such as pyramids or abstract forms). The game is designed to enhance spatial reasoning and problem-solving skills while providing an engaging and educational experience.

Game Mechanism

1. **Puzzle Structure:**
   - **Levels:** The game is divided into levels, each presenting a unique target shape that players must recreate using the seven Soma pieces.

    - **Progression:** The first level starts with the well-known 3×3×3 cube. As levels advance, the target shapes become more complex, adding new constraints and requiring innovative piece arrangements.
2. **Piece Manipulation:**
    - **Selection:** Players choose a Soma piece from the piece tray.
    - **Rotation:** Each piece can be rotated along the x, y, and z axes. Initially, rotation controls are hidden and are revealed only when the user accesses the tutorial option.
    - **Placement:** After rotating, the player drags the piece onto the 3D grid (game board). Visual drop indicators and snapping ensure that pieces align correctly on the board.
    - **Validation:** When a target shape is completed, the game validates the solution and advances the player to the next level.
3. **Scoring & Timing:**
    - A timer tracks the duration for each level.
    - Scoring is based on time taken and moves made, with bonus points for efficient solutions.

User Interface (UI) Design

1. **Main Activity Area (Game Board):**
    - **3D Grid:** Displays the puzzle board in a fixed default camera view, showing x, y, and z axes to simulate depth.
    - **Target Shape Preview:** A visual reference of the target shape (e.g., the classic cube for level one) is integrated into the interface, so players know what to aim for.
    - **Visual Aids:** Grid lines, shadows, and highlights help the user discern depth and correctly align pieces.
2. **Toolbar:**
    - **Activity Icon:** Indicates the active game session.
    - **Piece Tray:** Displays the seven Soma pieces for quick selection.
    - **Tutorial Button:** When clicked, this reveals guidance on how to rotate and place pieces. Rotation controls appear contextually when needed.
    - **Stop Button:** Allows players to exit the current game.

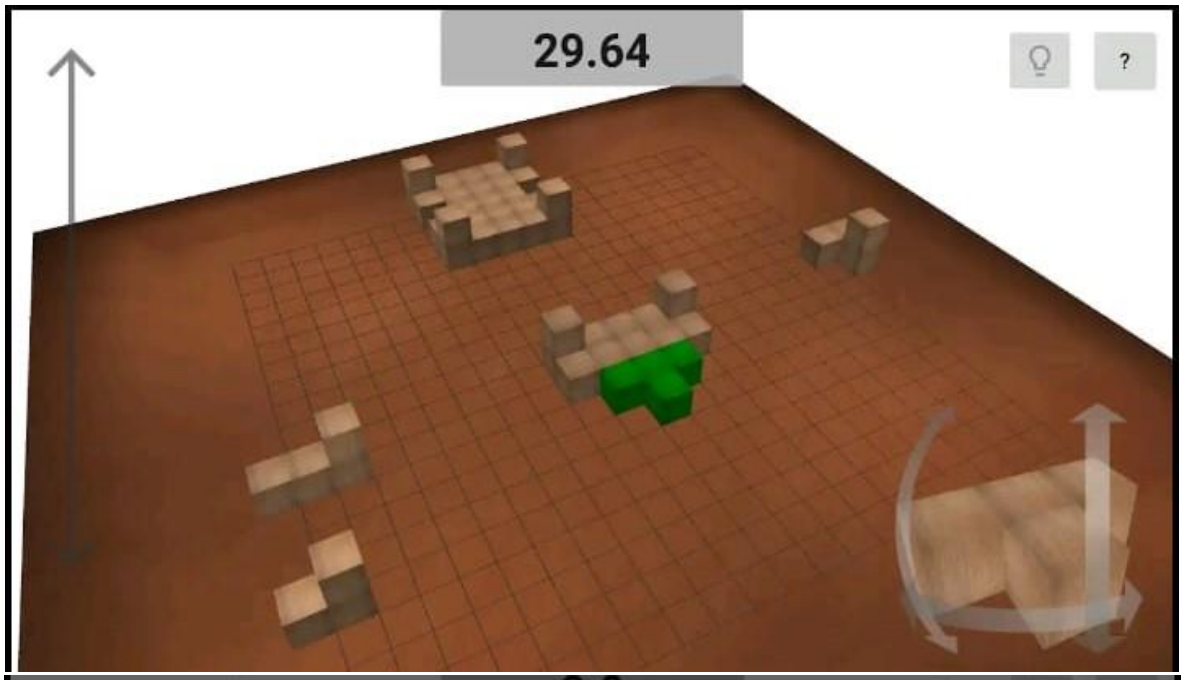
Controls

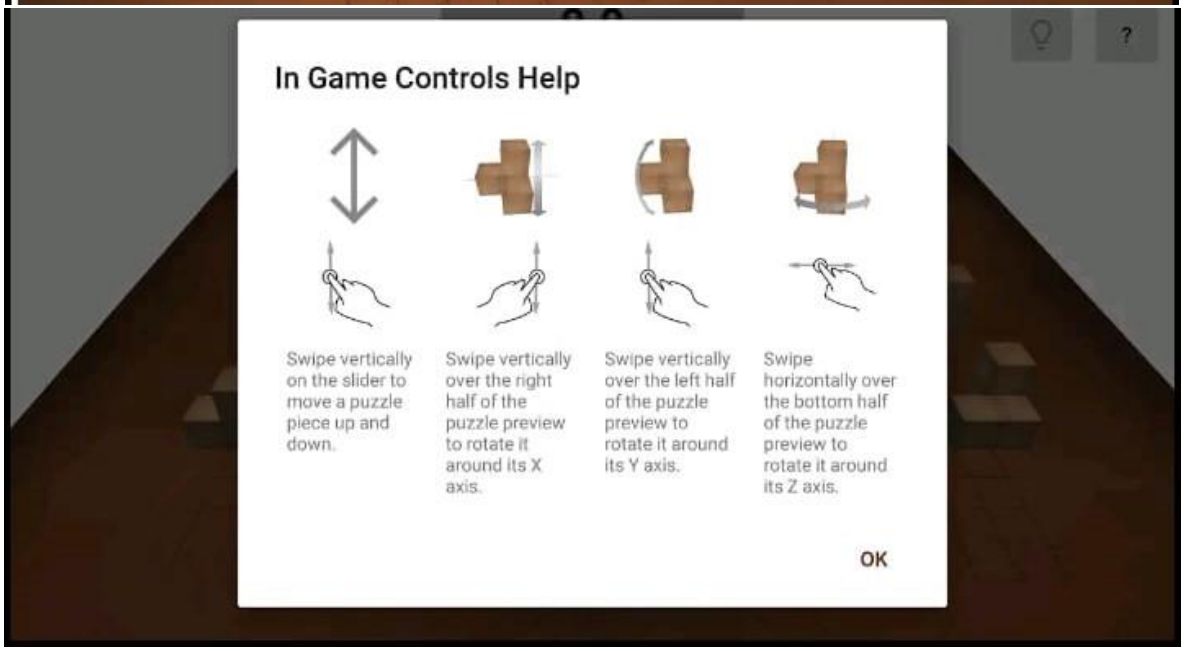
1. **Piece Manipulation Controls:**
    - **Selection & Drag-and-Drop:** The player selects a piece from the toolbar and drags it into the game board.
    - **Rotation:** Player can rotate the selected piece in x, y, z axis.
    - **Snap and Align:** As pieces approach valid grid positions, visual indicators and snapping features help ensure accurate placement.

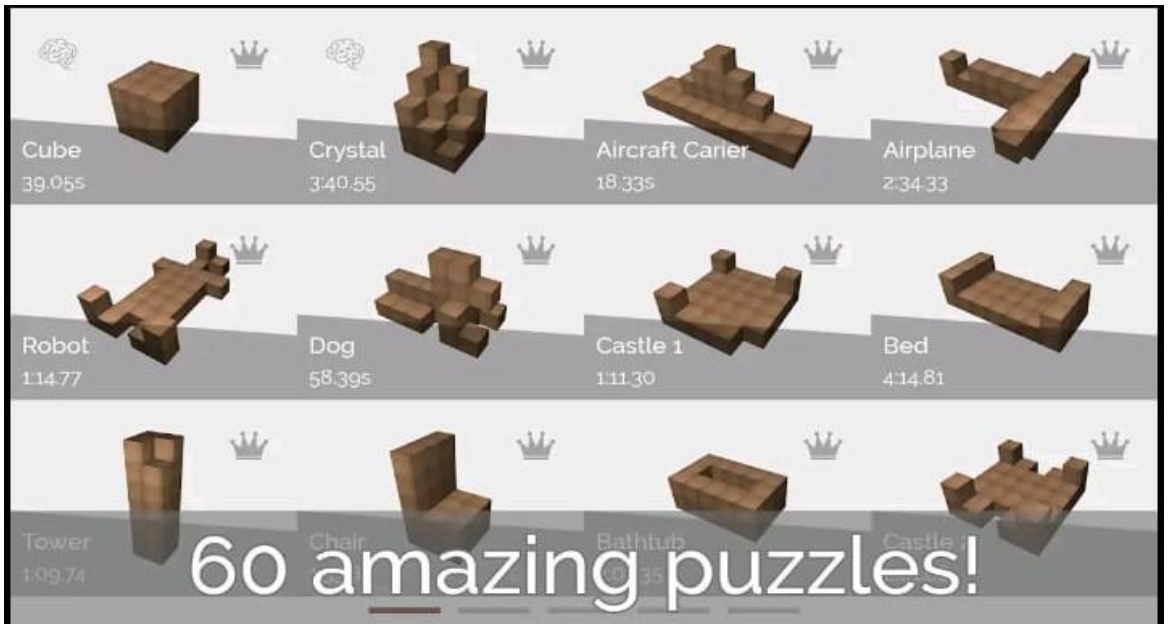
2. **Interface Navigation:**
    - **Fixed Camera:** The camera view is fixed by default, so players do not have to worry about adjusting the perspective.
    - **Toolbar Navigation:** Simple buttons in the toolbar allow players to access tutorials, view their piece tray, or stop the game.

Reference: https://play.google.com/store/apps/details?id=sk.martinflorek.somacube

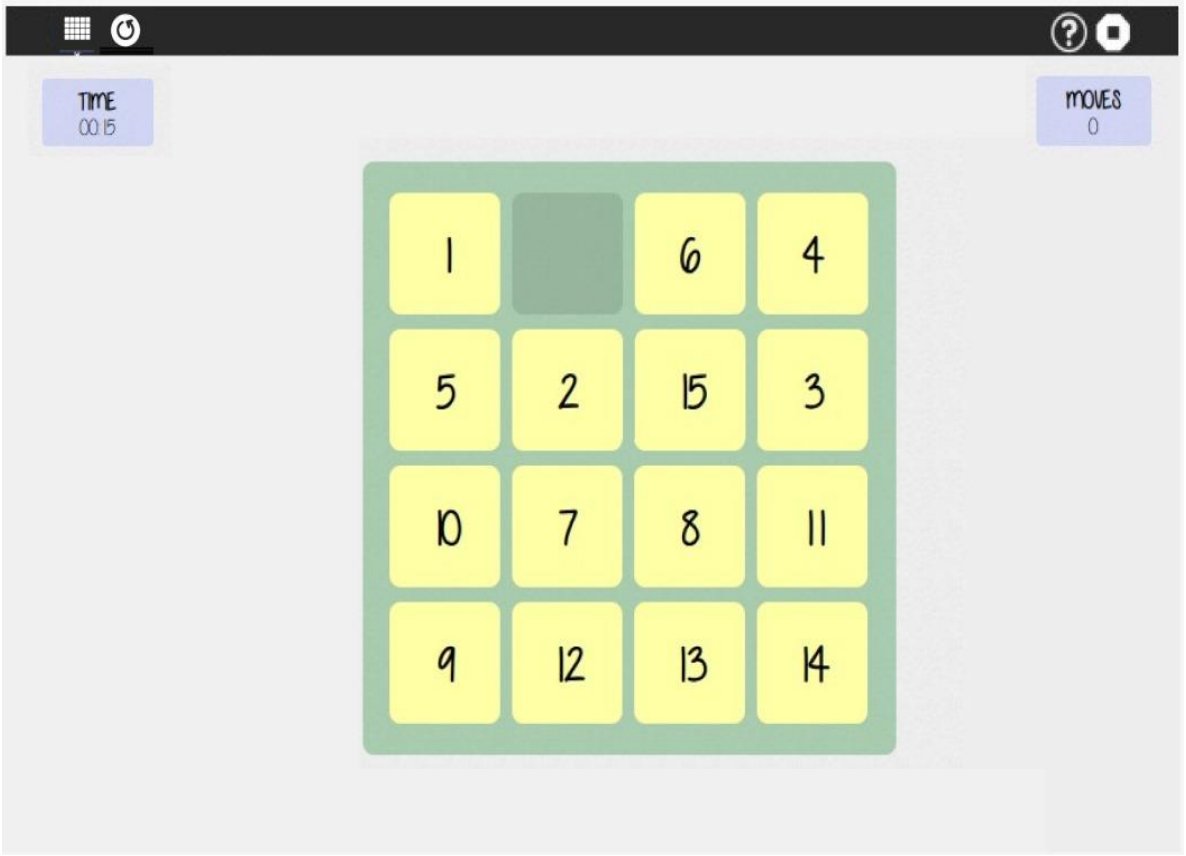
Visualizations:







4. Fifteen Puzzle



Overview:

The Fifteen Puzzle is a classic sliding puzzle consisting of a 4×4 grid with 15 numbered tiles (1–15) and one blank space. The objective is to rearrange the tiles into sequential order (reading row by row), using the blank space to slide adjacent tiles.

Game Initialization:

- **Solvable Puzzle Generation:**
  To ensure that each puzzle is solvable, the puzzle is generated by starting with the solved configuration and executing a series of random legal moves (only sliding tiles adjacent to the blank space). In each move, we randomly select from available moves, optionally excluding the reverse of the previous move, to effectively scramble the puzzle. This guarantees that every generated puzzle is solvable, as it is derived from the solved state through legal, reversible moves.
- **Grid Setup:**
  A 4×4 grid is initialized where each cell contains a numbered tile or remains empty. Tiles are presented in bright, appealing colors with clear, legible fonts.

User Input and Tile Movement:

- **Input Methods:**
  Players interact with the puzzle using a mouse, touch input, or keyboard. Only tiles adjacent to the blank space (horizontally or vertically) can be moved.
- **Movement Mechanics:**
  When a valid tile is selected, it slides smoothly into the empty space with an animated transition. Illegal moves (attempting to move non-adjacent tiles) are ignored.

Game Logic and Win Condition:

- **Move Validation:**
  After each move, the game updates the current configuration and checks if the tiles are arranged in numerical order with the blank in the bottom-right corner.
- **Victory Trigger:**
  Once the puzzle is solved, a victory screen is displayed, featuring celebratory animations and game statistics such as total moves and elapsed time.

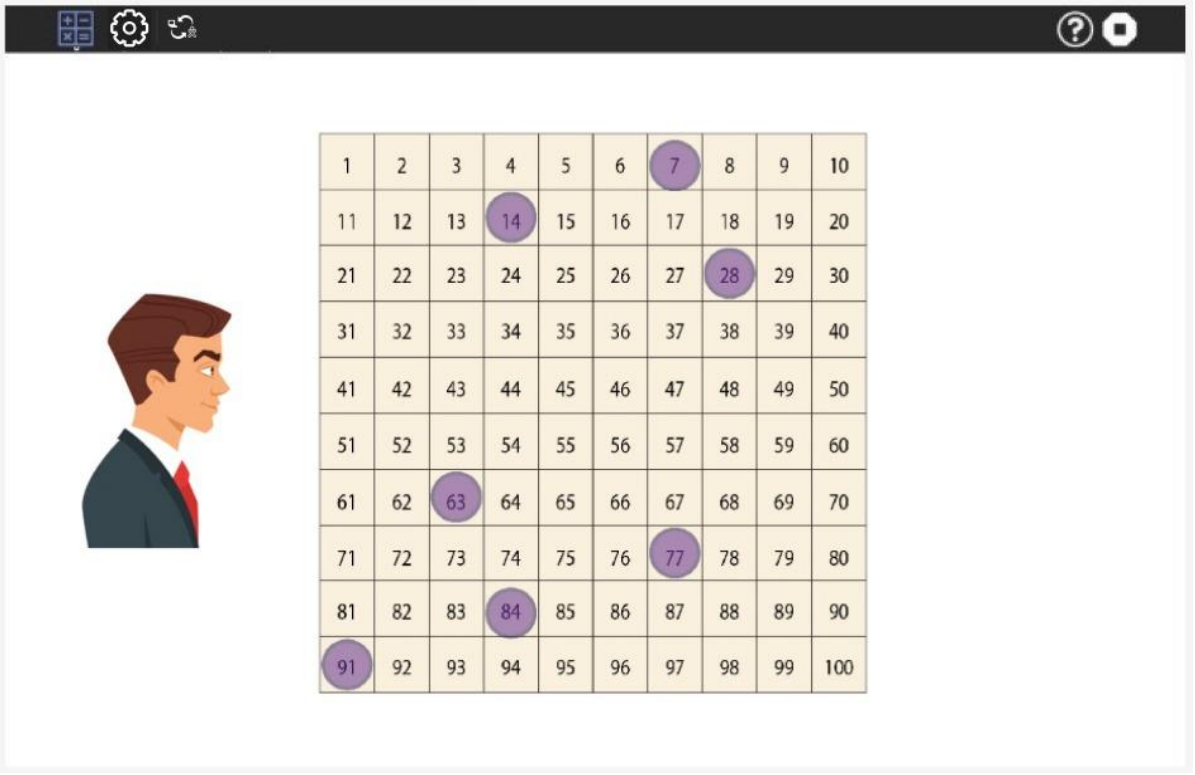
Additional Features:

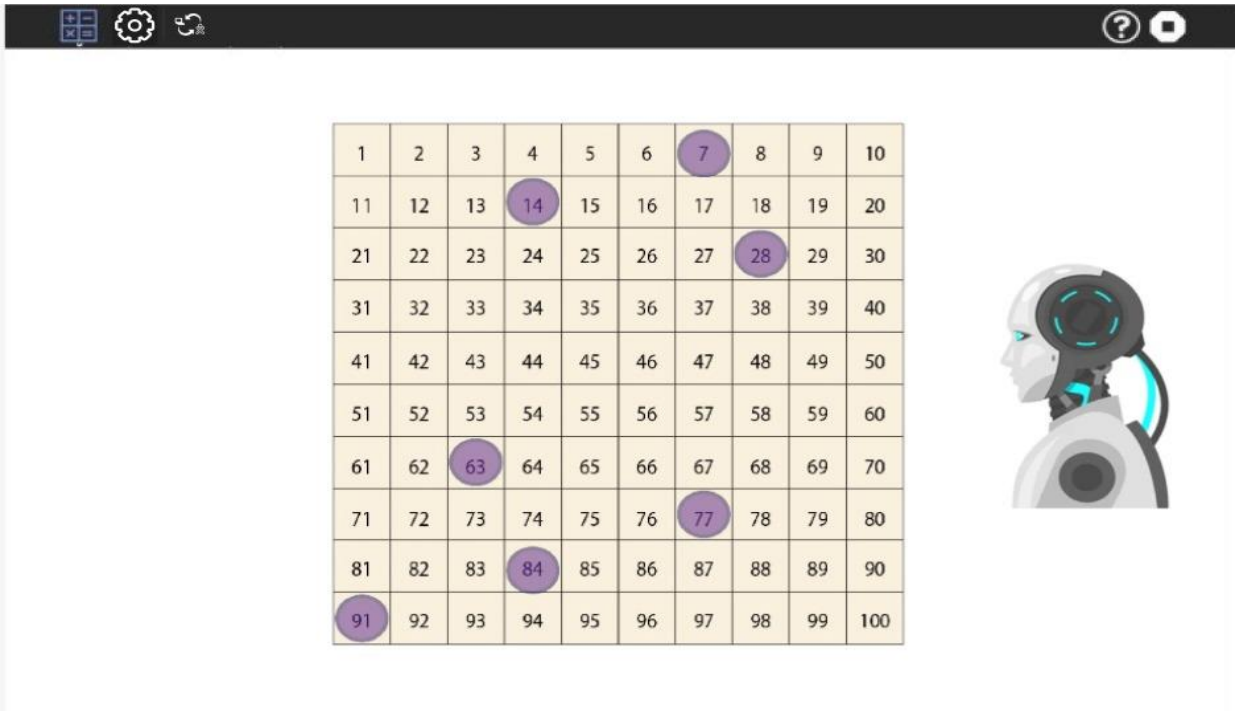
- **Move Counter and Timer:**
  The game tracks the number of moves made and includes a timer to add a competitive element. Players can challenge themselves to achieve a lower move count or complete the puzzle faster.
- **Reset and Shuffle Options:**
  Players can reset the puzzle back to the solved state or generate a new,

randomized, yet solvable puzzle. This feature encourages repeated play and practice.

- **Responsive Feedback:**
  Every valid move is accompanied by subtle sound effects and visual cues, such as a brief highlight or shadow effect on the moving tile, to provide immediate feedback. This responsive feedback enriches the interactive experience and helps players stay engaged.

5. Euclid's Game





Overview:

Euclid's Game is a two-player mathematical puzzle that brings to life the principles behind Euclid's algorithm. It not only challenges players to think strategically about each move but also demonstrates how repeated subtraction reveals the greatest common divisor (gcd) of two numbers.

How It Works:

The game starts by placing two distinct positive numbers on the board (for example, 15 and 6). These numbers are displayed on a customizable grid (ranging from 5×5 to 10×10) that organizes and shows all available moves.

- **Turn-Based Moves:**
  Players take turns selecting any two numbers already on the board and subtracting the smaller from the larger. For instance, if the board shows 15 and 6, subtracting 6 from 15 yields 9, which is then added to the board.
- **Building the Board:**
  As the game progresses, each new move adds another number to the board. Continuing our example:
  - Starting with **15** and **6**, the first move gives **15 – 6 = 9**, so the board becomes {15, 6, 9}.
  - On the next turn, subtracting **9 – 6 = 3** (a number not yet on the board) results in {15, 6, 9, 3}.
  - A subsequent move, such as **15 – 3 = 12**, then expands the board to {15, 6, 9, 3, 12}.
  - Since the gcd of 15 and 6 is 3, the board will eventually include all multiples of 3 up to 15 (i.e., 3, 6, 9, 12, and 15). When every possible new number appears, no further moves can be made.
- **Winning Condition:**
  The goal is to be the player who makes the last move. When a player cannot produce a new number because any subtraction yields an already present number, that player loses. In our example, after the complete set {3, 6, 9, 12, 15} is formed, the player who made the final move wins.

Strategic Insights:

The underlying strategy is determined by the total number of distinct moves available, which can be calculated as:

$$\frac{N}{\gcd(N, M)}$$

where N is the larger starting number and M the smaller. For example, with 15 and 6, this gives:

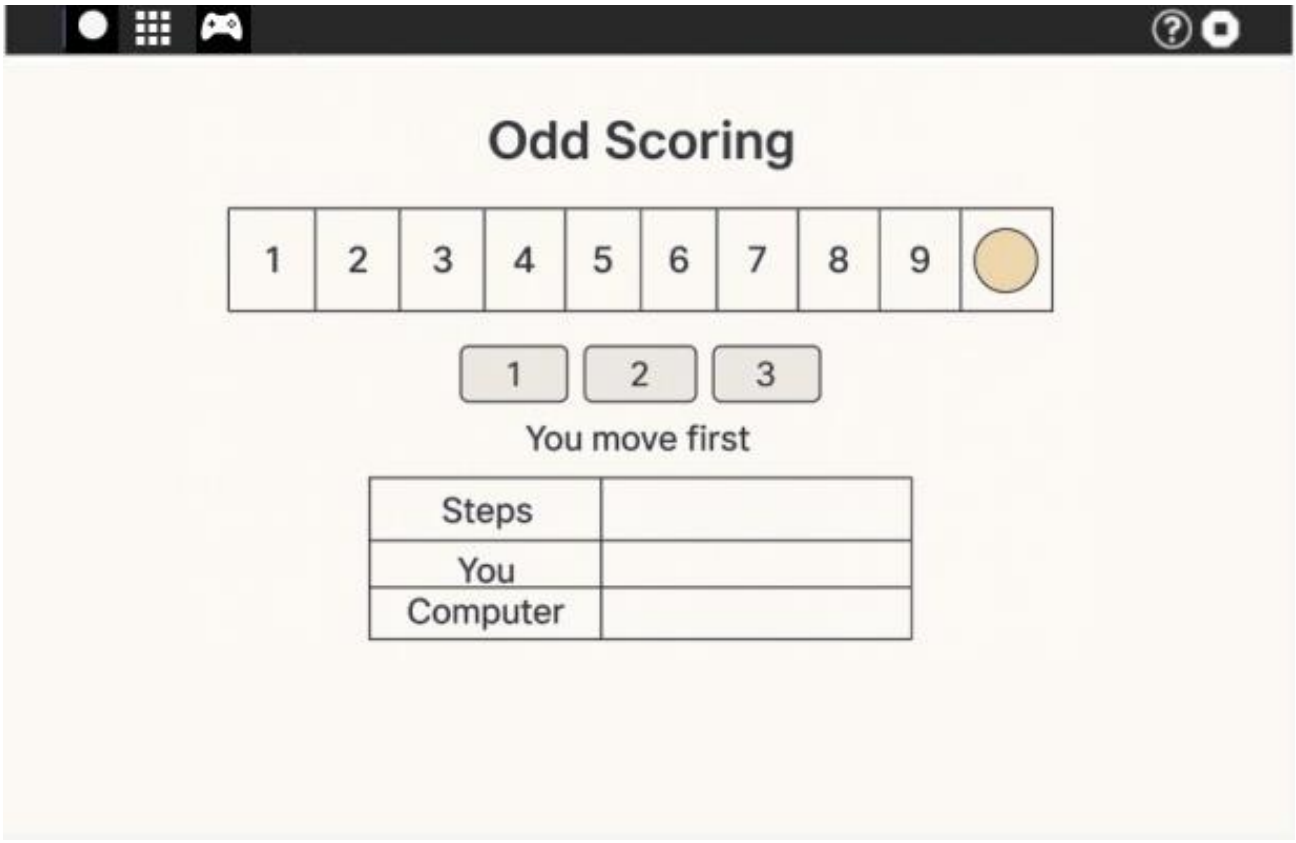
$$\frac{15}{3} = 5.$$

This means there will be 5 distinct numbers on the board, implying a total of 3 moves after the initial setup (since you start with two numbers). If this quotient is odd, the first mover, who makes the first subtraction, can force a win with perfect play. If it were even, the advantage would shift to the second player. By understanding this relationship, players can appreciate why choosing certain starting numbers or adjusting turn order (via a “Switch Player” option) can critically affect the outcome.

User Interface:

- **Grid Size Customization:**
  Having this option available in the settings lets players choose a grid size from 5×5 to 10×10. This flexibility enables them to tailor the game to their preferences—a 5×5 grid is perfect for a faster game with fewer moves, while a 10×10 grid supports a longer, more complex play experience.
- **Turn Order Control:**
  Incorporating a switch-player option gives the human player the ability to choose who goes first. This is particularly important because, as explained above, the mathematical ratio $\frac{N}{\gcd(N,M)}$ can predetermine the winning advantage. For example, since 15 always yields an odd quotient regardless of the second number, a player might opt to move second to counterbalance this edge.

6. Odd Scoring



Overview

The Odd Scoring Game **is a** two-player mathematical puzzle that blends strategy and arithmetic. It challenges players to think ahead and control the parity (odd/even nature) of their total score while moving a chip **across a** horizontal grid**.**

Game Mechanics

1. **Board Layout:**
    - **Grid:** The game is played on a horizontal grid (or "band") consisting of N cells, numbered from 1 (leftmost) to N (rightmost).
    - **Starting Position:** A chip starts at the rightmost cell (cell N).
2. **Turn-Based Moves:**
    - **Allowed Moves:** On each turn, players can move the chip leftwards by 1, 2, or 3 cells.
    - **Objective:** The game ends when the chip reaches the leftmost cell (cell 1).
3. **Scoring:**
    - Each move adds its numerical value (i.e., the number of cells moved) to that player's running total.
    - Because the total steps required are fixed at N−1 (an odd number), one player's total will be even and the other's odd.
4. **Winning Condition:** The winner is determined by the parity of each player's total steps: the player with an even total wins.
5. **Winning Strategy:**
    - A key strategy involves forcing the game into positions where the remaining steps form a multiple of 4. This pairing strategy enables a player to control the parity of their own total and secure a win with optimal play.

User Interface (UI)

1. **Toolbar Bar:**
    - The top of the screen features a toolbar with two options: Game Mode, which allows players to select the difficulty level (Easy or Hard), and Grid Size, which lets players adjust the number of cells (default is 10, adjustable between 2 and 10).
2. **Grid Display:**
    - A horizontal row of numbered cells is presented below the toolbar. Each cell is neatly bordered and clearly labeled.
    - A visual chip/circle indicates the current position on the grid.
3. **Interactive Controls:**
    - Move buttons labeled "1," "2," and "3" allow the player to choose how many cells to move the chip leftward.

- A scoreboard displays the running total of moves for both the player and the computer in real time.

Difficulty Modes:

Difficulty Modes integrated into the settings allow users to adjust the mode according to their skill level. Here how they work:

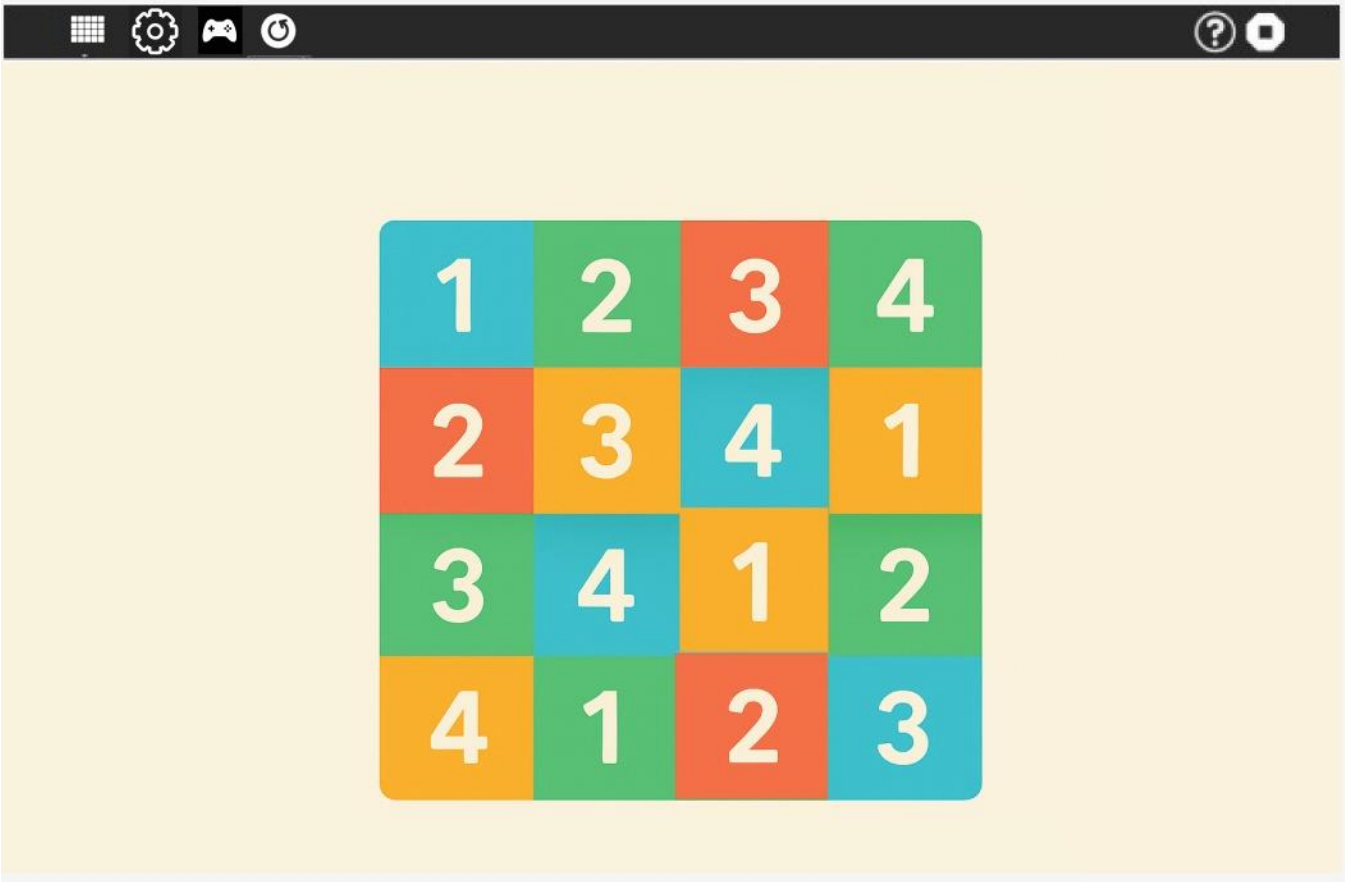
- **Hard Mode:** The computer opponent always plays optimally by calculating the best move that forces the remaining steps into a multiple of 4, ensuring a challenging experience.
- **Easy Mode:** While the computer still calculates the optimal move, it intentionally makes suboptimal choices with a preset probability (e.g., 30%), offering a more forgiving challenge for beginners.

Example Walkthrough

Consider a grid with 10 cells (thus the chip/circle must move a total of 9 cells):

- **Initial Setup:**
  - The chip starts at cell 10.
  - Total steps required: 10 −1 = 9
- **Your First Move:**
  - If you move 1 step, the chip moves from cell 10 to cell 9, and 8 steps remain.
  - This choice is strategic because 8 is a multiple of 4.
- **Computer's Turn and Your Response:**
  - Suppose the computer moves 2 steps (from cell 9 to cell 7).
  - To complete the pair to 4, you then move 2 steps (from cell 7 to cell 5).
  - At this point, the total steps you have made sum up to 1 + 2 = 3, while the computer has made 2, with 4 steps remaining.
- **Final Moves:**
  - If the computer now moves 3 steps (from cell 5 to cell 2), you are forced to move the final 1 step (from cell 2 to cell 1).
  - Final totals become: your total = 1 + 2 + 1 = 4 (even) and the computer's total = 2 + 3 = 5 (odd).
  - Since you have an even total, you win.

7. Make an Identity



Overview:

The Make an Identity game is an interactive puzzle designed to engage users with a classic mathematical concept. In a Make an Identity, players must arrange a set of distinct symbols in an n×n grid so that each symbol appears exactly once in every row and every column. This activity not only serves as an entertaining puzzle but also promotes logical reasoning, pattern recognition, and spatial awareness. Additionally, the game offers different modes, such as the standard (simple) Make an Identity and a symmetric variant that requires mirror symmetry along the main diagonal, catering to various skill levels and learning goals.

Game Mechanism:

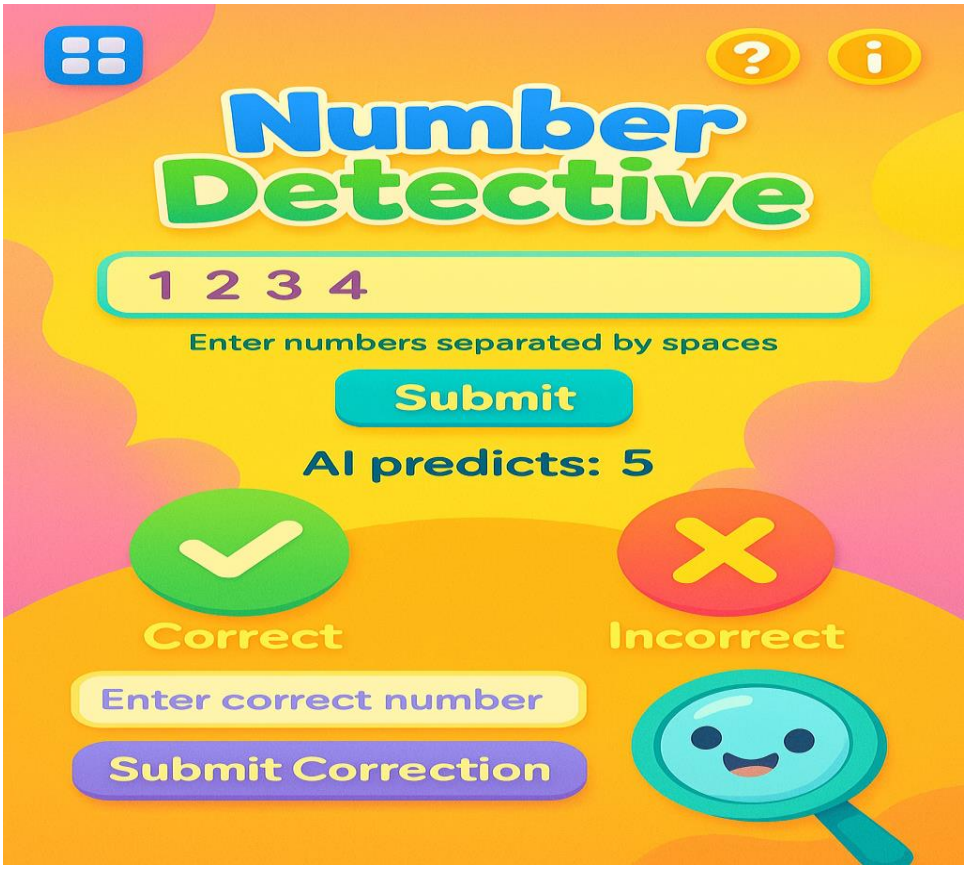
- **Core Puzzle Dynamics:**
  Players interact with the puzzle by rearranging symbols to satisfy the Make an Identity constraints. Two main manipulation methods are provided:
  - **Row Cycling:** Clicking adjacent to a row cycles its symbols left or right, enabling a quick shift in the order.
  - **Cell Swapping:** Clicking two cells within the same row swaps their contents, offering precision for fine-tuning arrangements.
- **Gameplay Modes:**
  - Simple Mode*:* Enforces the basic rule that each symbol must appear once per row and column.

    - Symmetric Mode*:* In addition to the standard rule, the grid must be symmetric across the main diagonal (i.e., the symbol at cell (i, j) must equal the one at (j, i)).
- **Grid Size Flexibility:**
  Players can select from multiple grid sizes (e.g., 3×3 for beginners, 4×4 as the default moderate challenge, and 6×6 for advanced puzzles). Changing the grid size dynamically adjusts the puzzle's difficulty and layout.

User Interface (UI):

- **Toolbar and Controls:**
  The UI features an intuitive toolbar that houses all key customization and control options. This includes:
    - A game mode option for selecting the gameplay mode (Simple or Symmetric).
    - A grid size selector allowing users to choose between 3×3, 4×4, and 6×6 grids.
    - A reset button to reinitialize or scramble the current puzzle without changing the user's settings.
- **Default Layout:**
  The game board defaults to a 4×4 numbered grid, offering a clear and straightforward visual presentation right from the start. This 4×4 board serves as the initial configuration, making it easy for users to understand the puzzle's structure immediately.

8. Number Detective



Overview

Number Detective is an engaging, educational math game designed to teach pattern recognition and introduce basic AI concepts. In this activity, players enter a sequence of numbers, and the game's AI predicts the next number in the sequence. If the prediction is incorrect, the player provides the correct number as feedback, which is used to refine the system's predictions over time. The game uses a hybrid approach combining simple rule-based logic with a lightweight machine learning model (e.g., linear regression or decision trees) to continuously improve its performance.

Game Mechanism

- **User Input:**
  Players enter a number sequence using a clearly labeled input field (e.g., "Enter numbers separated by spaces").
- **AI Prediction:**
  The system analyzes the sequence using:
  - **Rule-Based Algorithms:** To detect common patterns (such as arithmetic or geometric progressions).
  - **Simple Machine Learning Models:** Like linear regression or decision trees, which learn from historical corrections and refine predictions.
- **Feedback Loop:**
  - If the prediction is incorrect, the game prompts the player to enter the correct number.
  - This correction is sent back to update the system, enhancing future predictions.
- **Learning Process:**
  With each cycle of prediction and correction, the AI refines its understanding of the patterns, demonstrating core reinforcement learning principles in a feedback-driven context.

Frontend (User Interface)

Integration with Sugarizer:

- The activity is built using modern web technologies. The game logic and rendering are handled by PhaserJS, while the interactive UI components and state management can be implemented using frameworks like React or VueJS. This combination ensures a responsive and engaging experience that aligns with Sugarizer's web-based architecture.

UI Layout and Components:

- **Toolbar:**
  - A toolbar at the top displays the activity's icon, help, end button.

- **Main Activity Area:**
  - **Input Field:**
    A large, clearly labeled text box with a placeholder (e.g., “Enter numbers separated by spaces”).
  - **Prediction Display:**
    A prominent label that shows the prediction (e.g., “AI predicts: 5”).
  - **Feedback Controls:**
    - “Correct” and “Incorrect” buttons.
    - A correction input field appears if the prediction is marked incorrect.
- **Visual Enhancements:**
  - Consistent styling is achieved through CSS, while smooth animations enhance user engagement.

Note: This is a tentative design of the UI, and further modifications and refinements will be made as the project evolves.

Event Handling and Responsiveness:

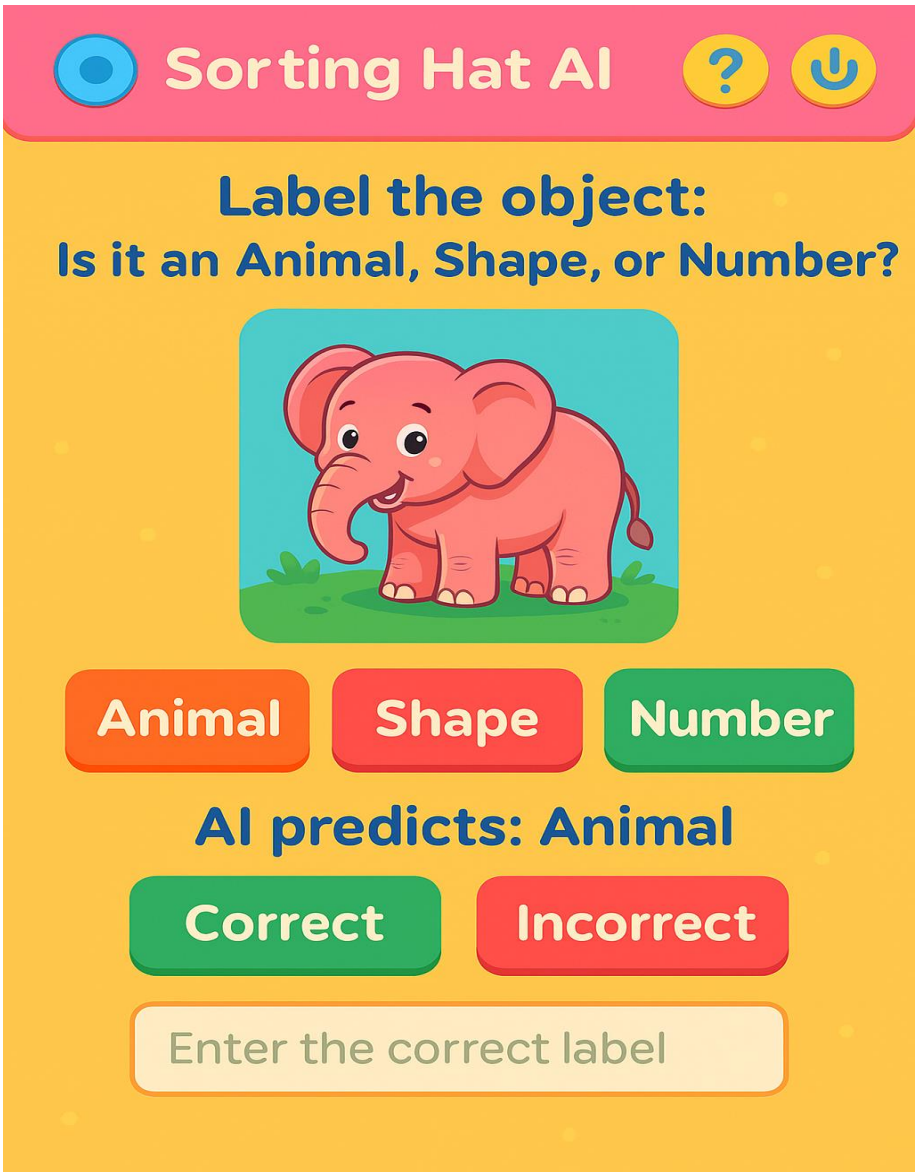
- Implement event listeners to capture user interactions (e.g., text modifications and button clicks).
- Use debouncing and throttling techniques to manage API calls efficiently, ensuring the UI remains responsive without overloading the backend.

Backend Integration

API Endpoints and Communication Flow:

- The backend is built using FastAPI (as part of the Sugar-AI module) and provides endpoints (e.g., /ask) specifically for the prediction functionality.
- **Workflow:**
  1. **Data Submission:**
     The frontend sends the entered number sequence to the FastAPI endpoint via an HTTP POST request.
  2. **Processing:**
     The backend processes the sequence using its rule-based logic combined with the simple machine learning model to generate a prediction.
  3. **Response:**
     The prediction is returned as a JSON response and displayed on the frontend.
  4. **Feedback Loop:**
     If the player marks the prediction as incorrect, the correction is sent back to the backend, updating the model and improving future predictions.

9. Sorting Hat AI



Overview

Sorting Hat AI is an engaging, educational game designed to teach how AI classifies objects. In this activity, players are presented with images of objects (animals, shapes, or numbers) and are asked to label them correctly. The game's AI learns from these labels using classification algorithms—specifically Decision Trees or k-Nearest Neighbors (k-NN)—and applies this knowledge to classify new objects. When the AI misclassifies an item, children provide the correct label, improving the model's accuracy over time. This interactive feedback loop demonstrates core machine learning principles in a fun, hands-on way.

Game Mechanism

- **User Interaction:**
  - **Object Presentation:**
    The game displays an image or illustration of an object (e.g., an animal, shape, or number) in the main activity area.
  - **User Labeling:**
    Players select the correct label from predefined categories (e.g., “Animal”, “Shape”, “Number”) using interactive buttons or icons.
- **AI Classification:**
  - The system processes the labeled examples using:
    - **Decision Trees:** To create a clear, interpretable classification structure.
    - **k-Nearest Neighbors (k-NN):** To classify new objects based on the similarity to labeled examples.
  - The AI then predicts the category of the displayed object, and the predicted label is shown on-screen.
- **Feedback Loop:**
  - If the AI’s prediction is correct, the game confirms the accuracy.
  - If the prediction is incorrect, the user is prompted to select the correct label.
  - This feedback is sent back to the system, updating the classification model and refining future predictions.
- **Learning Process:**
  - With every cycle of labeling and correction, the AI improves its decision boundaries, demonstrating how classification models can learn over time through user interaction.

Frontend (User Interface)

Integration with Sugarizer:

- The activity is built using modern web technologies. The game logic and rendering are handled by PhaserJS, while the interactive UI components and state management can be implemented using frameworks like React or VueJS. This combination ensures a responsive and engaging experience that aligns with Sugarizer’s web-based architecture.

UI Layout and Components:

- **Toolbar:**
  - A colorful toolbar at the top displays the activity’s icon, help button, and end button.
- **Main Activity Area:**
  - **Object Display:**
    A central area where the object image is shown.

    - **Label Options:**
      Colorful, child-friendly buttons or icons are provided for each category (e.g., “Animal”, “Shape”, “Number”).
    - **AI Prediction Display:**
      A prominent label shows the AI’s current prediction (e.g., “AI predicts: Shape”).
    - **Feedback Controls:**
        - “Correct” and “Incorrect” buttons enable the player to indicate if the prediction is right.
        - If marked “Incorrect,” a correction interface appears for selecting the correct label.
- **Visual Enhancements:**
    - Consistent styling is achieved through CSS, while smooth animations enhance user engagement.
- **Event Handling and Responsiveness:**
    - Event listeners capture user interactions (button clicks, selection events) to trigger classification requests.
    - Debouncing and throttling techniques ensure that API calls to update the model are managed efficiently without overwhelming the backend.

Note: This is a tentative design of the UI, and further modifications and refinements will be made as the project evolves.

Backend Integration

API Endpoints and Communication Flow:

- The backend is built using FastAPI (integrated as part of the Sugar-AI module) and provides endpoints (e.g., /classify) tailored to the classification functionality.
- **Workflow:**
    1. **Data Submission:**
       The frontend sends the object label data or the user-selected correction via an HTTP POST request to the FastAPI endpoint.
    2. **Processing:**
       The backend processes the labeled data using the chosen classification algorithm (Decision Trees or k-NN) to update the model and generate a prediction for new objects.
    3. **Response:**
       The AI’s prediction is returned as a JSON response and displayed on the frontend.
    4. **Feedback Loop:**
       When the player corrects a misclassification, the correction is sent back to update the model, thereby improving future predictions.

Project Technology Stack

I will develop my Sugarizer math games using the following technologies:

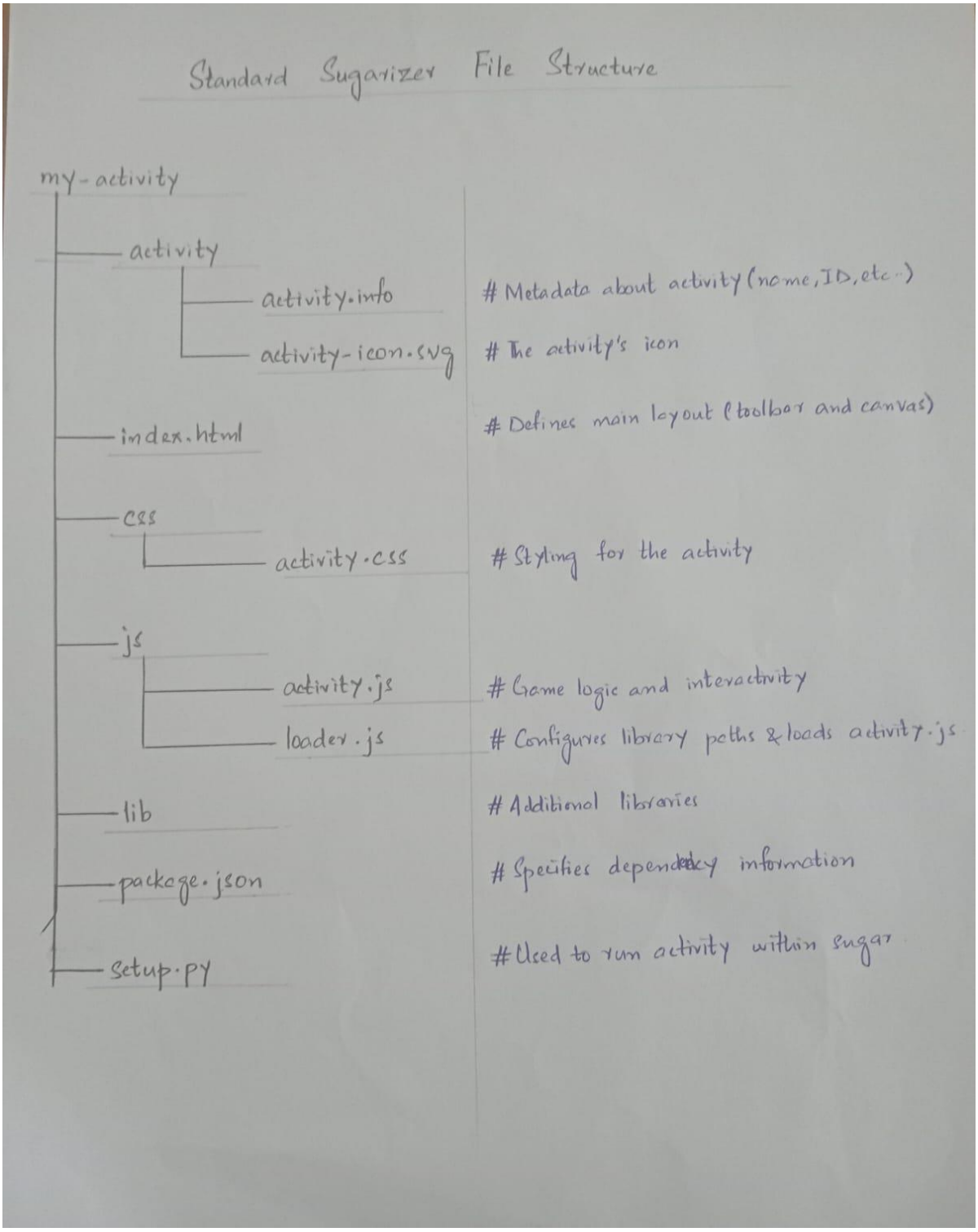
- **HTML5 & CSS3:**
  These will be used to create the structure and style the user interface. I will ensure a responsive design that adapts to different screen sizes and input methods (touch, mouse, and keyboard) in line with Sugarizer's UI requirements.
- **JavaScript:**
  JavaScript will be the primary language for implementing game logic and interactivity. I will use modern JavaScript (ES6+) to ensure code modularity and maintainability.
- **Game Framework (PhaserJS/ThreeJS):**
  Depending on the complexity of each math game, I plan to leverage:
  - **PhaserJS** for 2D game development
  - **ThreeJS** for any 3D components.
    This will enable smooth animations, physics, and asset management while ensuring a rich gaming experience.
- **UI/State Management:**
  For games that require more complex UI elements or robust state management, I will integrate a modern JavaScript framework such as React or VueJS. These frameworks facilitate building component-based architectures and handling dynamic data interactions.
- **Backend Technologies (if needed):**
  In cases where a backend is required—for instance, for real-time multiplayer features, I plan to use FastAPI to create REST endpoints that facilitate data synchronization and user authentication.

Sugarizer Activity Framework:

I will follow the Sugarizer development guidelines for all math games. This includes:

- **Activity Integration:** Ensuring the activity icon adapts its color when launched from the Sugarizer home view.
- **Journal Integration:** Using Sugarizer's Journal API to automatically save game progress and other session data.
- **Localization:** Implementing multilingual support to make the game accessible to a wider audience.

    - **Collaborative Features:** Ensuring that the activities are designed to be used by multiple users concurrently, following Sugarizer's collaboration patterns.
    - **Lifecycle Management:** Adhering to Sugarizer's guidelines for activity initialization, state management, and cleanup.
- Furthermore, my activity development will follow the standard Sugarizer file structure, which is organized as follows:



Development Approach:

1. **Prototype Design:**
   I will begin by designing visual mockups and wireframes for each math game (e.g., Four Color Map Game, Fifteen Puzzle, and AI-powered games like Number Detective). This will be done using tools such as Figma or Adobe Photoshop.
2. **Modular Development:**
   Each game will be developed as an independent Sugarizer activity. I will structure the code in modular components:
   - **UI Components:** Built with HTML5 and CSS3 to ensure consistency with Sugarizer's design language.
   - **Game Logic Modules:** Using JavaScript and the selected game framework (PhaserJS or ThreeJS) to implement core functionalities.
   - **Integration Layer:** A dedicated layer to interface with Sugarizer's APIs (for Journal, localization, and collaboration).
3. **Iterative Development and Testing:**
   I will follow an iterative development process, regularly testing the activities on different devices to ensure responsiveness, usability, and integration with the Sugarizer environment. This will involve:
   - Unit and integration tests for core functionalities.
   - Manual testing of activity lifecycle events and integration with the Sugarizer Journal.
   - Gathering feedback from Sugar Labs mentors and community members through pull requests and code reviews.
4. **Documentation and Deployment:**
   Each activity will be documented thoroughly, including setup instructions, code comments, and user guides. I will also prepare a demo video for each game, showcasing gameplay and integration with Sugarizer, in accordance with Sugar Labs' guidelines.

Project Timeline

Community Bonding Period 8th May – 1st June	○ Implement game design and algorithms for different games. ○ Create player flowcharts for all the games. ○ Maintain regular updates with mentors and seek possible modifications and feedback on any game.
Week 1 2nd June – 8th June	○ **Tasks:** ○ Develop the basic UI (toolbar, game board, color palette). ○ Implement core mechanics: interactive coloring and solution verification (“Check” button). ○ **Outcome:** ○ A working prototype of the Four Color Map Game.
Week 2 9th June – 15th June	○ **Tasks:** ○ Refine the Four Color Map Game (animations, bug fixes, additional levels). ○ Begin the Broken Calculator: design the

	calculator UI and implement basic equation input (with the disabled key). ○ **Outcome:** ○ A polished Four Color Map Game and a basic prototype for Broken Calculator.
Week 3 16th June - 22th June	○ **Tasks:** ○ Complete equation validation, score updating, and bonus logic for Broken Calculator. ○ Perform initial testing and debugging. ○ **Outcome:** ○ A fully functional Broken Calculator activity.
Week 4 17th June - 23rd June	○ **Tasks:** ○ Develop and finalize the 3D grid with full piece selection, drag-and-drop, and rotation controls. ○ Create and integrate all target shapes (starting with the 3×3×3 cube and any additional levels planned). ○ Implement snapping functionality, visual aids (e.g., grid lines, shadows), and complete game logic.

	○ Conduct thorough testing and debugging to ensure smooth 3D interactions and a polished user experience. ○ **Outcome:** ○ A complete, fully functional Soma Cubes activity ready for user interaction.
Week 5 30th June - 6th July	○ **Tasks:** ○ Develop the 4×4 grid, implement solvable puzzle generation, and code tile movements with animations. ○ Add a move counter, timer, and victory condition detection. ○ **Outcome:** ○ A functional prototype of the Fifteen Puzzle.
Week 6 7th July - 13th July	○ **Tasks:** ○ Refine the Fifteen Puzzle (UI enhancements, bug fixes, improved feedback). ○ Begin Euclid's Game development: set up the game board and implement basic

	subtraction-based, turn-based play. ○ **Mid-Evaluation:** Present completed Four Color Map, Broken Calculator, Soma Cubes, and Fifteen Puzzle for mentor feedback. ○ **Outcome:** ○ A polished Fifteen Puzzle, a basic prototype of Euclid's Game, and documented feedback from mid-evaluation.
Week 7 14th July - 20th July	○ **Tasks:** ○ Refine Euclid's Game with smooth turn transitions and accurate move tracking. ○ Initiate Odd Scoring development: design the horizontal grid, implement chip movement (1, 2, or 3 cells), and set up real-time score tracking. ○ **Outcome:** ○ A stable Euclid's Game and a working prototype of Odd Scoring.
Week 8 21th July – 27th July	○ **Tasks:** ○ Enhance Odd Scoring with improved UI feedback and further debugging. ○ Begin development of Make An Identity: set up

	the activity interface, implement logic for identity creation (for example, combining digits or equations to form an identity), and design the core gameplay flow. ○ **Outcome:** ○ A polished Odd Scoring game and a basic working version of Make An Identity.
Week 9 28th July – 3rd August	○ **Tasks:** ○ Complete and refine Make An Identity gameplay, including interactive elements and dynamic difficulty settings. ○ **Outcome:** ○ A fully functional Make An Identity activity.
Week 10 4th August - 10th August	○ **Tasks:** ○ Begin development of Number Detective: design the input interface and integrate initial rule-based prediction logic. ○ Set up FastAPI endpoints for AI predictions and feedback. ○ **Outcome:** ○ A prototype of

	Number Detective with basic AI functionality ready for further enhancement.
Week 11 11th August - 17th August	○ **Tasks:** ○ Enhance Number Detective by integrating lightweight machine learning and a robust feedback loop. ○ Initiate development of Sorting Hat AI: design the object display, interactive labeling, and implement initial classification logic (using Decision Trees or k-NN). ○ **Outcome:** ○ Advanced progress on Number Detective and a basic prototype of Sorting Hat AI.
Week 12 18th August - 24th August	○ **Tasks:** ○ Finalize Sorting Hat AI: polish the UI, optimize the feedback loop, and ensure accurate classification. ○ Integrate and test all nine activities within the Sugar environment; conduct final debugging and update documentation and demo materials. ○ **Outcome:**

	○ All nine games are fully integrated and refined, ready for final evaluation.
Final Evaluation 25th August – 1st September	○ By this time, successfully **at least 8 activities** will be developed for Sugar. ○ Submit the final evaluation for Google Summer of Code 2025, and all the documented work.

Note: Testing the activity will involve checking it on different screen sizes to ensure compatibility across devices. and examining edge cases and ensuring error-free performance. Defining the user flow, and scope of these games in the community bonding period will ensure smooth making of these activities within the coding period.

Regular meetings and feedback will be taken from mentors to ensure smooth working and making of these activities.

Availability

Sugar labs is the only organization I'm applying to and thus have no commitment during GSoC towards any other organization.

I have my end-of-semester exams scheduled to conclude by the end of May and after that I have no special commitments towards my college, no exams are scheduled during gsoc period therefore I will be able to dedicate approximately 35-40 hours weekly to the project.

Additionally, I will be available and fully reachable from **08:00 PKT to 23:00 PKT** (as per 24-hour format).

Post-GSoC Plan

After GSoC, I plan to continue contributing to Sugar Labs by maintaining and improving the math activities developed during the GSoC program. Additionally, I plan to mentor and guide future contributors who wish to contribute to Sugar Labs**.** Beyond this, I also aim to explore new AI-driven educational tools to further expand the community's impact on open-source education.